41장 타이머

41.1 호출 스케일링

- **호출 스케줄링**
  - 함수를 명시적으로 호출하지 않고 일정 시간이 경과된 이후에 호출되도록 함수 호출을 예약하려면 타이머 함수를 사용
- **타이머 생성 함수** : setTimout, setInterval
- **타이머 제거 함수** : clearTimeout, clearInterval

타이머 함수는 ECMAScript 사양에 정의된 빌트인 함수 ❌

- 하지만 브라우저 환경과 Node.js 환경에서 모두 전역 객체의 메서드로서 타이머 함수를 제공
  - 즉, 타이머 함수는 호스트 객체
- setTimout과 setInterval 모두 일정 시간이 경과된 이후 콜백 함수가 호출되도록 타이머를 생성]
- setTimout 함수가 생성한 타이머는 단 한 번 동작하고, setInterval 함수가 생성한 타이머는 반복 동작
- 자바스크립트 엔진은 단 하나의 실행 컨텍스트 스택을 갖기 때문에 두 가지 이상의 태스크를 동시에 실행할 수 없다.
  - 즉, 자바스크립트 엔진은 싱글 스레드로 동작한다.
  - 이런 이유로 타이머 함수 setTimeout과 setInterval은 비동기 처리 방식으로 동작한다.

41.2 타이머 함수

41.2.1 setTimeout/clearTimeout

- 두 번째 인수로 전달받은 시간(ms)으로 **단 한 번** 동작하는 타이머를 생성
- 이후 타이머가 만료되면 첫 번째 인수로 전달받은 콜백 함수가 호출된다.

```
const timeoutId = setTimeout(func|code[, delay, param1, param2, ...]);
```

매개변수	설명
func	타이머가 만료된 뒤 호출될 콜백 함수 ※ 콜백 함수 대신 코드를 문자열로 전달할 수 있다. 이때 코드 문자열은 타이머가 만료된 뒤 해석되고 실행된다. 이는 흡사 eval 함수와 유사하며 권장하지는 않는다.
delay	타이어 만료 시간(밀리초(ms) 단위). setTimeout 함수는 delay 시간으로 단 한 번 동작하는 타이머를 생성한다. 인수 전달을 생략한 경우 기본값 0이 지정된다. ※ delay 시간이 설정된 타이머가 만료되면 콜백 함수가 즉시 호출되는 것이 보장되지는 않는다. delay 시간은 태스크 큐에 콜백 함수를 등록하는 시간을 지연할 뿐이다. 태스크 큐는 42장 "비동기 프로그래밍"에서 자세히 살펴볼 것이다. ※ delay가 4ms 이하인 경우 최소 지연 시간 4ms가 지정된다.
param1, param2, ...	호출 스케줄링된 콜백 함수에 전달해야 할 인수가 존재하는 경우 세 번째 이후의 인수로 전달할 수 있다. ※ IE9 이하에서는 콜백 함수에 인수를 전달할 수 없다.

```
// 1초 후 타이머가 만료되면 콜백 함수가 호출된다.
setTimeout(() => console.log('Hi'), 1000);

// 1초 후 타이머가 만료되면 콜백 함수가 호출된다.
// 이때 콜백 함수에 'Lee'가 인수로 전달된다.
setTimeout((name) => console.log(`Hi ${name}`), 1000, 'Le
e');

// 타이머가 취소되면 setTimeout 함수의 콜백 함수가 실행되지 않는다.
clearTimeout(timerId);
```

41.2.2 setInterval/clearInterval

- 두 번째 인수로 전달받은 시간(ms)으로 **반복** 동작하는 타이머를 생성
- 이후 타이머가 만료될 때마다 첫 번째 인수로 전달받은 콜백 함수가 반복 호출된다.

```
const timerId = setInterval(func|code[, delay, param1, param2, ...]);
```

```
let count = 1;

// 1초 후 타이머가 만료될 때마다 콜백 함수가 호출된다.
```

```
// setInterval 함수는 생성된 타이머를 식별할 수 있는 고유한 타이머 id를
const timeoutId = setInterval(() => {
    console.log(count); // 1 2 3 4 5
    // count가 5이면 setInterval 함수가 반환한 타이머 id를 clearInt
    // 타이머를 취소한다. 타이머가 취소되면 setInterval 함수의 콜백 함수
    if(count++ === 5) clearInterval(timeoutId);
}, 1000);
```

41.3 디바운스와 스로틀

- scroll, resize, input, mousemove 같은 이벤트는 짧은 시간 간격으로 연속해서 발생
- 이러한 이벤트에 바인딩한 이벤트 핸들러는 과도하게 호출되어 성능에 문제를 일으킬 수 있다.
- 디바운스와 스로틀 : 짧은 시간 간격으로 연속해서 발생하는 이벤트를 그룹화해서 과도한

  이벤트핸들러의 호출을 방지하는 프로그래밍 기법

41.3.1 디바운스

- 짧은 시간 간격으로 이벤트가 연속해서 발생하면 이벤트 핸들러를 호출하지 않다가 일정 시간이 경과한 이후에 이벤트 핸들러가 한 번만 호출
  - 즉, 디바운스는 짧은 시간 간격으로 발생하는 이벤트를 그룹화하여 마지막에 한 번만 이벤트 핸들러가 호출
  - resize 이벤트 처리, input 요소에 입력된 값으로 ajax 요청하는 입력 필드 자동완성 UI 구현, 버튼 중복 클릭 방지 처리 등에 유용하게 사용
  - Underscore의 debounce 함수나 Lodash의 debounce 함수를 사용하는 것을 권장

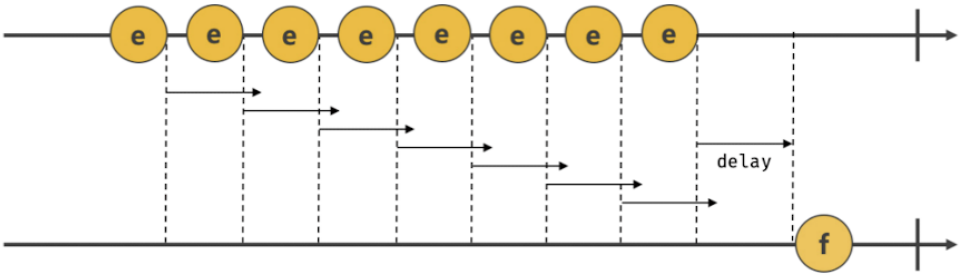

그림 41-2 디바운스

41.3.2 스로틀

- 스로틀은 짧은 시간 간격으로 이벤트가 연속해서 발생하더라도 일정 시간 간격으로 이벤트 핸들러가 최대 한 번만 호출
- 즉, 스로틀은 짧은 시간 간격으로 연속해서 발생하는 이벤트를 그룹화해서 일정 시간 단위로

  이벤트 핸들러가 호출되도록 호출 주기를 만든다.
  - scroll 이벤트 처리, 무한 스크롤 UI 구현 등에 유용하게 사용
  - Underscore의 throttle 함수나 Lodash의 throttle 함수를 사용하는 것을 권장

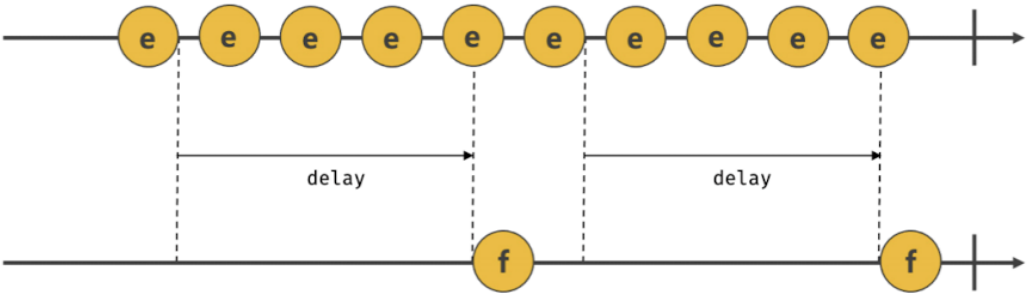

그림 41-3 스로틀